Submission Sheet 1

Danny Heinrich Matthias Kasperidus Leonard Kleinhans

29. October 2014

1 Exercise 1.1: McCulloch-Pitts Neural Net

a) A two layered McCulloch-Pitts Neural Net is sufficient to model the given boolean function $f(x) = \bar{x}_1\bar{x}_2\bar{x}_3 \vee x_1x_2 \vee x_2x_3$. Each conjunction is modelled by a neuron, which only fires if all three inputs are active. The disjunction neuron fires as soon as one conjunction neuron fires. See Figure 4

b) As seen in a) it is sufficient to use as two layered McCulloch-Pitts Neural Net to model the given boolean function g(x). The $\wedge$ binds stronger than the $\vee$ and therefore $g(x)$ is equivalent to $g(x) = x_1x_{2x3} \vee x_1x_2\bar{x}_4 \vee x_1x_4\bar{x}_2$. See Figure 5

2 Exercise 1.2: Perceptron Classifier

Apparently the given quadrangle is convex. So one can calculate the weight vectors $\vec{w_1}, \vec{w_2}, \vec{w_3}, \vec{w_4}$ by deriving the four hyper planes. It is easy to get the hyperplanes in the form $\vec{x} = \vec{p} + r\vec{q}$ with $\vec{x}, \vec{p}, \vec{q} \in R^2$ $r \in R$. This form is equivalent to the normalform $(\vec{x} - \vec{p})\ \vec{n} = 0$ with normal vector $\vec{n}$, which can also be written as $\vec{x}\ \vec{n} = \vec{p}\ \vec{n}$ and is almost what we want. See Figure 6 for the Quadrangle and Figure 7 for the constructed net. We now will calculate the four hyperplanes H_{ab}, H_{bc}, H_{cd} and H_{da} :

$$H_{ab} : \vec{x} = \vec{a} + r(\vec{b} - \vec{a}) = \begin{pmatrix}1\\1\end{pmatrix} + r\left(\begin{pmatrix}2\\-2\end{pmatrix} - \begin{pmatrix}1\\1\end{pmatrix}\right) = \begin{pmatrix}1\\1\end{pmatrix} + r\begin{pmatrix}-1\\-3\end{pmatrix}$$

$$\Leftrightarrow H_{ab} : 3x_1 + x_2 = 4 \text{ with } \vec{n} = \vec{w_1} = \begin{pmatrix}3\\1\end{pmatrix}$$

$$H_{bc} : \vec{x} = \vec{b} + r(\vec{c} - \vec{b}) = \begin{pmatrix}2\\-2\end{pmatrix} + r\left(\begin{pmatrix}0\\-1\end{pmatrix} - \begin{pmatrix}2\\-2\end{pmatrix}\right) = \begin{pmatrix}0\\-1\end{pmatrix} + r\begin{pmatrix}-2\\1\end{pmatrix}$$

$$\Leftrightarrow H_{bc} : (-1)x_1 + (-2)x_2 = 2 \text{ with } \vec{n} = \vec{w_2} = \begin{pmatrix}-1\\-2\end{pmatrix}$$

$$H_{cd} : \vec{x} = \vec{c} + r(\vec{d} - \vec{c}) = \begin{pmatrix}-1\\1\end{pmatrix} + r\left(\begin{pmatrix}0\\-1\end{pmatrix} - \begin{pmatrix}-1\\1\end{pmatrix}\right) = \begin{pmatrix}-1\\1\end{pmatrix} + r\begin{pmatrix}-1\\-2\end{pmatrix}$$

$$\Leftrightarrow H_{cd} : 2x_1 + (-1)x_2 = -3 \text{ with } \vec{n} = \vec{w_3} = \begin{pmatrix}2\\-1\end{pmatrix}$$

$$H_{da} : \vec{x} = \vec{d} + r(\vec{a} - \vec{d}) = \begin{pmatrix}-1\\1\end{pmatrix} + r\left(\begin{pmatrix}1\\1\end{pmatrix} - \begin{pmatrix}-1\\1\end{pmatrix}\right) = \begin{pmatrix}-1\\1\end{pmatrix} + r\begin{pmatrix}2\\0\end{pmatrix}$$

$$\Leftrightarrow H_{da} : 2x_2 = 2 \text{ with } \vec{n} = \vec{w_4} = \begin{pmatrix}0\\2\end{pmatrix}$$

Because $(0, 0)$ is inside the qudrangle, we can use that to get the inequalities:

$$\begin{aligned} 3x_1 + x_2 &\leq 4 \\ -1x_1 + -2x_2 &\leq 2 \\ 2x_1 + -1x_2 &\geq 2 \\ 2x_2 &\leq 2 \end{aligned}$$

3 Exercise 1.3: Perceptron Learning

As on can see in Figures 1 to 3 the perceptron learning algorithm correctly classifies only the third data set displayed in Figure 3. It shows that a singe layer perceptron learning algorithm can not classify every set correctly in such way that the algorithm terminates for given condition that every example must be classified correctly. In fig. 3 one can see that precision and recall are close to 1 meaning that nearly all examples are classified correctly. Only a few false classified examples remain. This leads to the conclusion that the seperating hyperplane for the whole set can not be constructed. A small error remains in every iteration of the algorithm. In figure 2 the precision rate drops drastically compared to figure 1 and 3. which induces a greater error. Finally the algorithm managed to determinate a weight vector to seperate the whole data set. Since every example is correctly classified there are no more errors and thus precision and recall converge to 1.

To run the code first import the function with R-Studio by using the source() function or running the code inside R-Studio. After the function has been imported correctly it can be called in the R console taking 2 parameters. Perceptron(path,maxInit) One possible approach would be :

source("path/to/source/code/name.R)

Perceptron("path/to/data/set1.txt",maxIterations) with maxIterations taking positive integers, respresenting the maximum iterations.

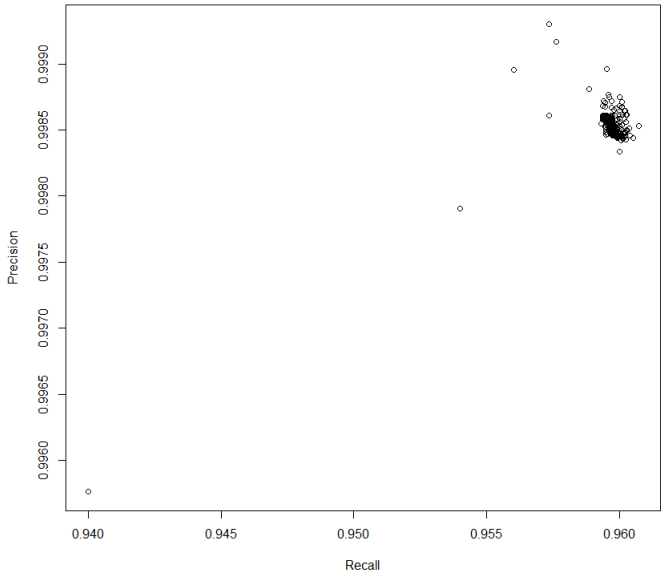

Figure 1: Precision Recall Curve of Set 1.

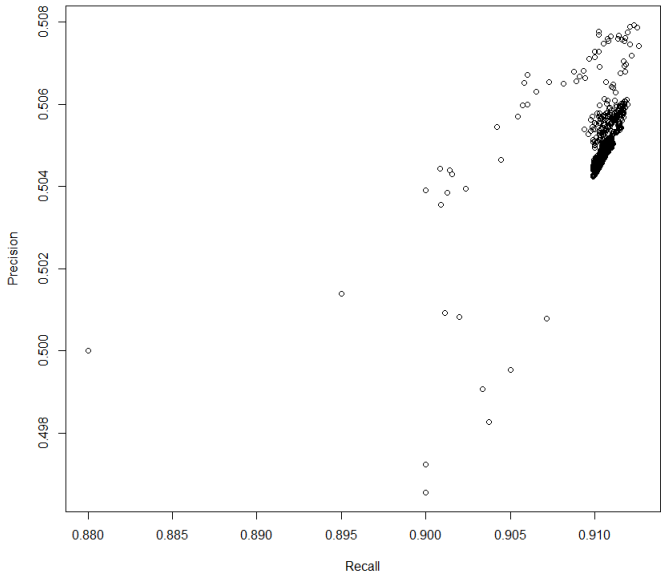

Figure 2: Precision Recall Curve of Set 2.

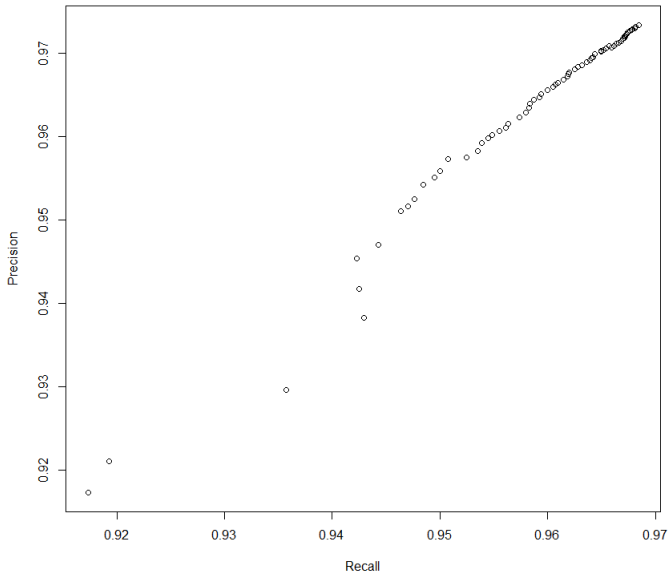

Figure 3: Precision Recall Curve of Set 3.

4 Exercise 1.4: Classification into more than two classes

One option could be, to train for N classes n ANN's: Every ANN classifies if the input is in class $n \in N$ or not. Basically one would iterate over $\mathcal{P}(N)$ in such way that $X \in \mathcal{P}(N)$ contains all and only all classes which were correctly classified for a given input. Furthermore one would test if there is more than one set meeting this condition. The algorithm would return the set with the most classes in it.

Another possibility is to use two layered Perceptron classifiers to classify the input regarding a tuple of classes. After finding the correct class the algorithm would then take that class as another input and tuple it with a class, which has not been used yet. As soon as all classes are processed by the algorithm the last correct class would be the one where the input would be most likely in.

Another option are 2^n classifiers, one for every pair of classes. The algorithm would then return every class out of the tuple in which the input lies. The union over all returned classes would then suffice for a correct classification.

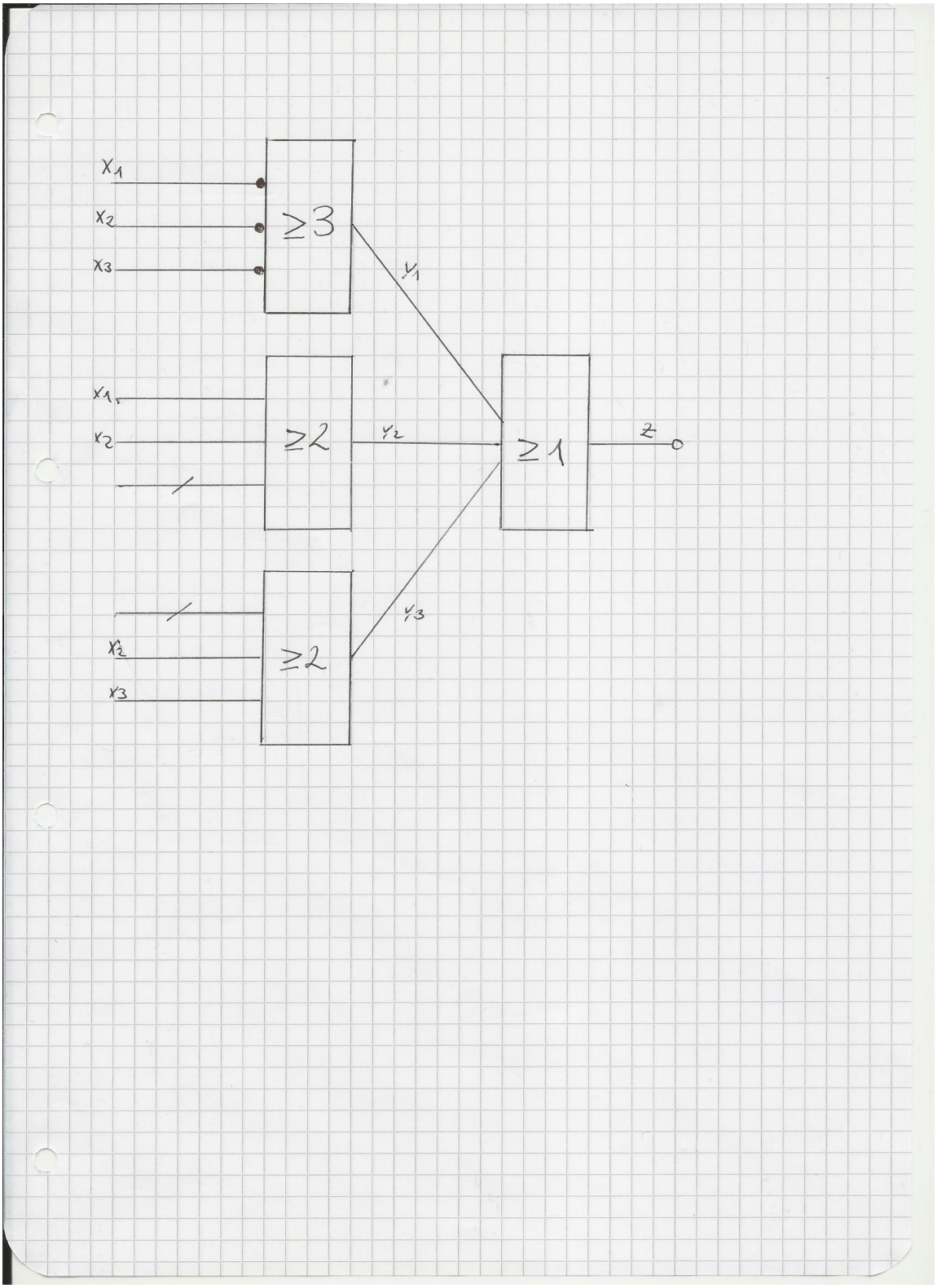

Figure 4: McCulloch-Pitts Neural Network a

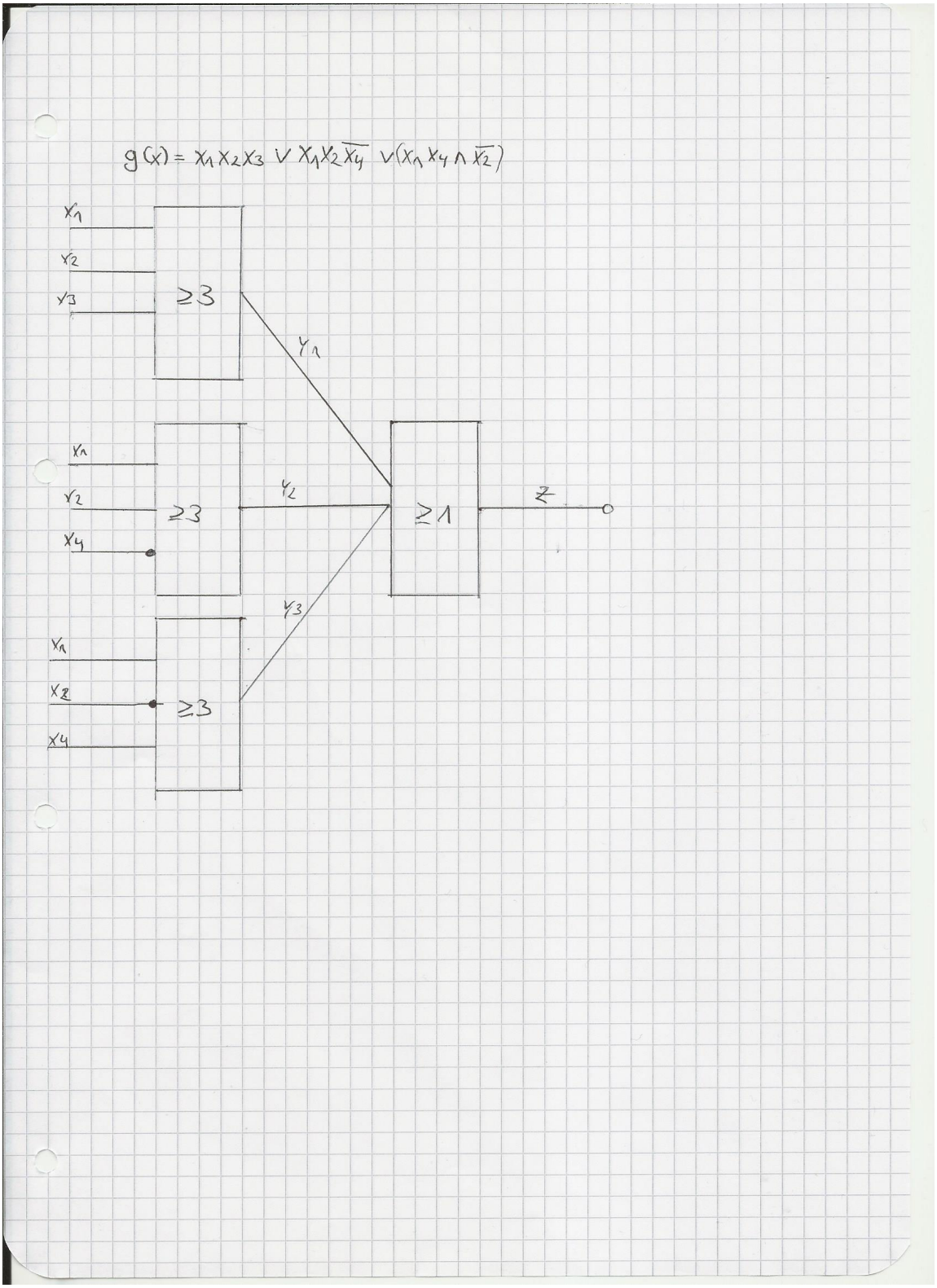

Figure 5: McCulloch-Pitts Neural Network b

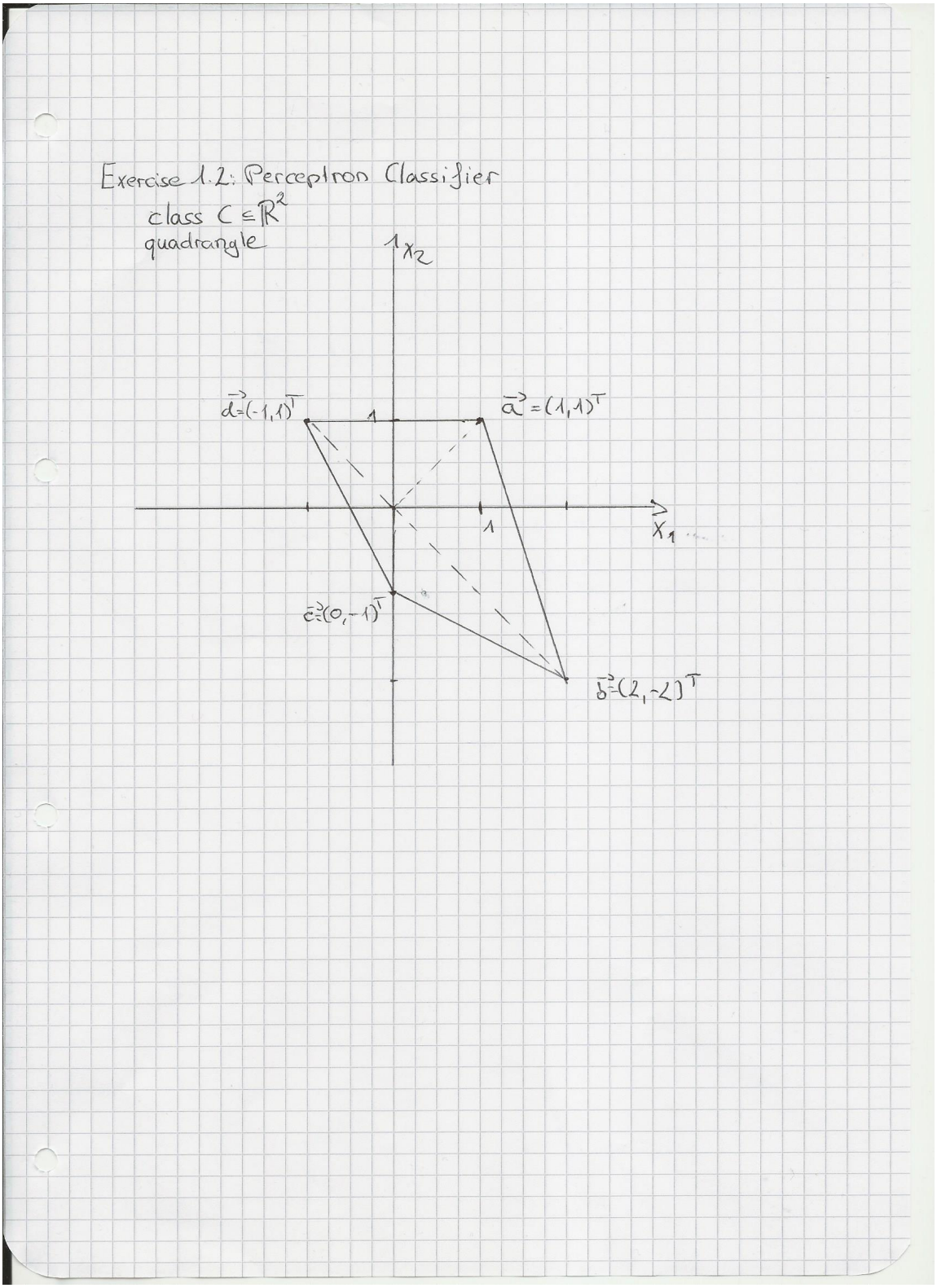

Figure 6: Quadrangle

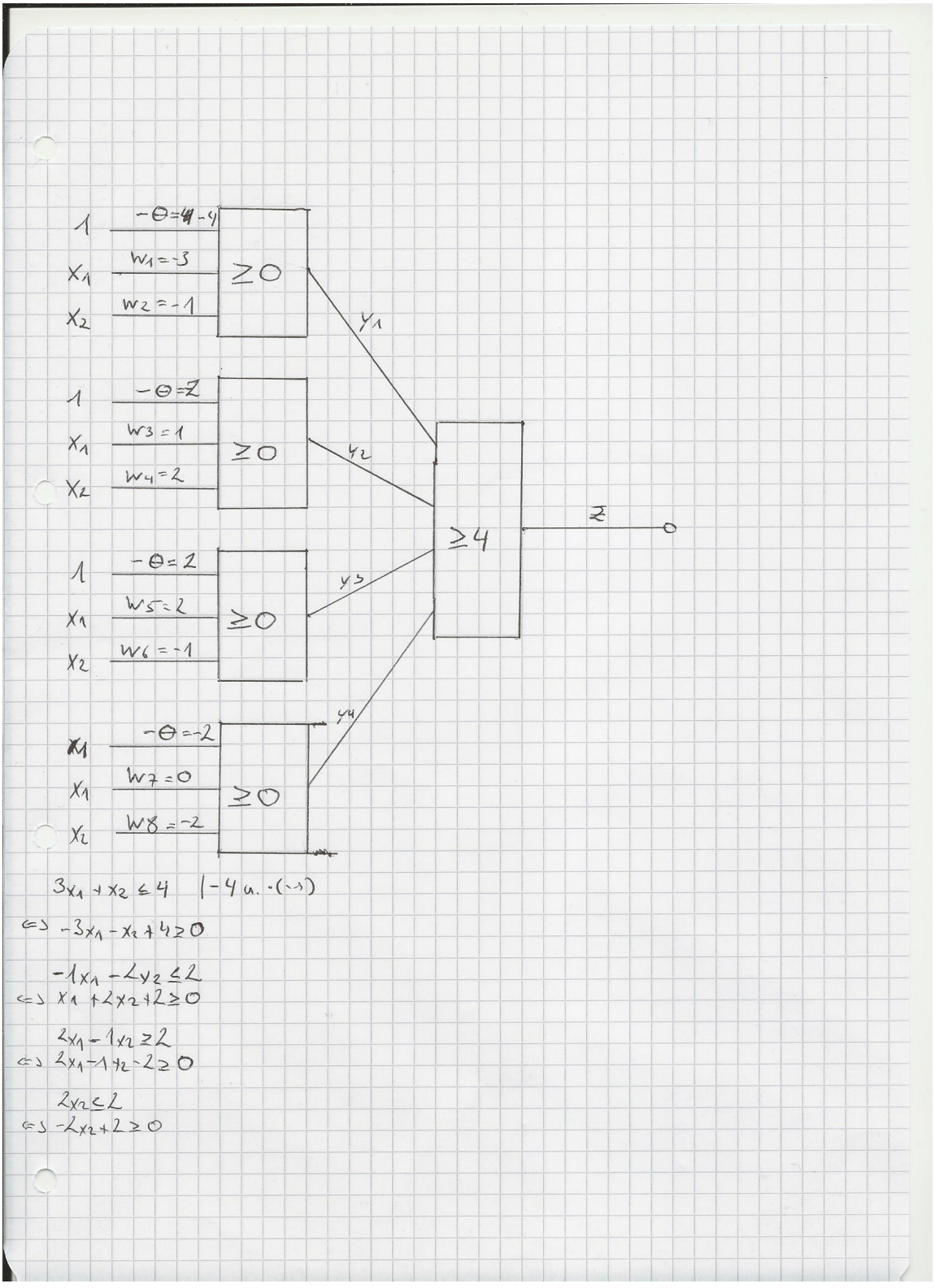

Figure 7: Perceptron Net Ex 1.2